Lecture 6
Primality, Factoring, RSA, Hensel's Lemma

CRT and the number of solutions - we have a congruence

$$a_k x^k + a_{k-1}x^{k-1} + \cdots + a_0 \equiv 0 \pmod{n}, \quad a_i \in \mathbb{Z}$$

We want to know all solutions mod n, and in particular the number of solutions. Write $n = p_1^{e_1} p_2^{e_2} \dots p_r^{e_r}$. Then solving the congruence mod m reduces to solving it mod $p_i^{e_i}$ for all i. If x satisfies the congruence mod n then it is a solution of the congruence mod $p_i^{e_i}$ for all i. Conversely if x_1 is a solution mod $p_1^{e_1}$ and x_2 is a solution mod $p_2^{e_2}$, etc., then the CRT says that there exists a unique $x \mod n$ such that $x \equiv x_i \mod p_i^{e_i}$ for all i.

Now for this x, $x \equiv x_i \mod p_i^{e_i}$, so

$$\begin{aligned} a_k x^k + a_{k-1}x^{k-1} + \cdots + a_0 &\equiv a_k x_i^k + a_{k-1}x_i^{k-1} + \cdots + a_0 \\ &\equiv 0 \pmod{p_i^{e_i}} \end{aligned}$$

Therefore x is a solution mod n, so this process gives a bijection.

$$\left\{\begin{matrix}\text{solutions} \\ \text{mod } p_1^{e_1}\end{matrix}\right\} \times \left\{\begin{matrix}\text{solutions} \\ \text{mod } p_2^{e_2}\end{matrix}\right\} \cdots \left\{\begin{matrix}\text{solutions} \\ \text{mod } p_r^{e_r}\end{matrix}\right\} \leftrightarrow \left\{\begin{matrix}\text{solutions} \\ \text{mod } n\end{matrix}\right\}$$

In particular, the total number of solutions mod n is

$$\prod_{i=1}^{r} \#\text{ of solutions mod } p_i^{e_i}$$

Primality Testing: Given n, we want to determine if n is prime or composite. Input n has $\log n$ digits. We call an algorithm efficient if it's polynomial in input - in this case, poly($\log n$) steps. Obvious algorithm is to divide by every prime starting from 2 to $\lfloor \sqrt{n} \rfloor$, which is $O(\sqrt{n})$ steps, or $\exp(\frac{1}{2}\log n)$.

Test using Fermat's Little Theorem - if n is prime and $n \nmid a$, then $a^{n-1} \equiv 1 \mod n$.

1. Pick an integer $a \in \{2 \dots n-1\}$.
2. Compute (a, n): if it is > 1, then n is composite.
3. Otherwise compute a^{n-1}: if $\not\equiv 1 \mod a$ then done, n is composite. If $\equiv 1 \mod n$, then is inconclusive. Could try another random a. There are composite numbers that satisfy $a^{n-1} \equiv 1 \mod n$ for all $(a, n) = 1$ (eg., 561) called Carmichael numbers.

Refinement: if $a^{n-1} \equiv 1 \mod n$, compute $a^{\frac{n-1}{2}}$ (since $n-1$ is even). If n is prime, must equal $\pm 1 \mod n$.

$$\begin{aligned} \text{If } a^{\frac{n-1}{2}} &\not\equiv \pm 1 \mod n, \text{ then } n \text{ is composite} \\ &\equiv -1 \mod n, \text{ then inconclusive, go to another } a \\ &\equiv 1 \mod n \text{ and } 4|n-1, \text{ then repeat with } a^{\frac{n-1}{4}} \end{aligned}$$

If n passes all these tests for a given a, n is a strong pseudoprime to base a. If n is prime, it's going to pass all these tests. If n is composite, it's pseudoprime to base a for at most $\frac{3}{4}$ of all possible a (usually much smaller).

So if we pick a random $a \mod n$ then n will pass the test with probability at most $\frac{3}{4}$. If it passes, then pick another random a. The probability of n passing k tests will be at most $(\frac{3}{4})^k$, which decreases exponentially with k. And so if you do $c \log n$ trials, the at most probability goes like $\frac{1}{n^*}$. So if n passes $c \log n$ trials (for some large enough $c \approx 100$), then probability that n is prime is very close to 1.

This is poly(log) steps, but we want a deterministic algorithm. Solved in 2002 by AKS (Agrawal, Kayal, Saxena). The main idea is that $n > 2$ is prime if and only if

$$(x-a)^n \equiv x^n - a \pmod n \quad \text{as polynomials}$$

Check different values of a, but there are n possible choices of a and expansion is slow. Way to avoid is with CRT by looking at both sides mod n and modulo a small degree polynomial.

Factorization If n is composite, how do we factor in poly($\log n$) time. The obvious way is to divide by all, which is $O(\sqrt{n})$.

Pollard Rho Let $f(x) = x^2 + 1$. $x_0 = 1, x_1 = f(x_0) \mod n, x_n = f(x_{n-1}) \mod n$ gives the sequence $x_0, x_1, \ldots x_i$. Heuristic: if $n = pm$ with small p then this sequence will start repeating mod p earlier than mod n. So if $x_j \equiv x_i \mod p$ but not mod n, then $(x_i - x_j, n)$ will give a factor of n.

In practice only need to compute x_k and x_{2k}, x_{2k} and x_{4k}, etc., and check gcd. Heuristic: if p is smallest prime factor of n, we expect to get a factor of n using this algorithm in about $\sqrt{p}$ steps.

Elliptic curve based factoring gives $\exp(c\sqrt{\log n \log\log n})$. Number field sieve gives $\exp(c(\log n(\log\log n)^2)^{\frac{1}{3}})$. We still don't have an efficient algorithm for factoring, a fact that much of modern cryptography is based on.

Cryptography - RSA Alice and Bob (A and B) want to pass messages, and Carol is eavesdropper. A can send a message to B, converted into bits - equivalent to sending an integer m. Wants to encrypt the number so C can't understand it.

Obvious method is to use a shared key model, where A and B have some shared key. With a message m, A can send $m + k$, or $m \oplus k$ (where $\oplus$ is the bitwise exclusive OR). B can decrypt the message by subtracting k: $(m + k) - k = m$, $(m \oplus k) \oplus k = m$. This is not so good if we want to send multiple messages - if C sees $m_1 + k$ and $m_2 + k$, then C can figure out $m_1 - m_2$, which gives her some information about m_1 and m_2, which is bad. This is also not efficient (number of keys needed grows quadratically as number of members to pass messages between increases).

Instead, use public key cryptography. One model is RSA (Rivest, Shamir, Adleman). The idea is that B generates two large primes p and q of about equal size. Set $N = pq$.

$$\phi(N) = \phi(p)\phi(q) = (p-1)(q-1)$$

Then B chooses e coprime to $\phi(N)$ and computes $f = e^{-1} \mod \phi(N)$. B publishes N and e as his public key. If A wants to send message m, assuming that $0 < m < N$ (if not, then we break into chunks), and $(m, N) = 1$. A then computes $m^e \mod N$ and sends it to B, who decrypts it by computing $(m^e)^f \mod N$. The idea is that

$$\begin{aligned} f &= e^{-1} \\ fe &\equiv 1 \pmod{\phi(N)} \\ &= 1 + k\phi(N) \\ m^{ef} &= m^{1+k\phi(N)} \\ &= m(m^{\phi(N)})^k \\ &\equiv m1^k \pmod{N} \\ &\equiv m \pmod{N} \end{aligned}$$

f is secret, so C has no way to compute m from m^e - this relies on the hardness of factoring.

Hensel's Lemma - this is a way to solve congruences mod p^e if we know solutions mod p (analog to Newton's Method for finding roots of polynomials).

Theorem 26 (Hensel's Lemma)**.** *Suppose that $f(x) \in \mathbb{Z}[x]$, $f(a) \equiv 0 \mod p^j$, and $f'(a) \not\equiv 0 \mod p$. Then there's a unique $t \mod p$ such that $f(a + tp^j) \equiv 0 \mod p^{j+1}$. That is, there's a unique solution b mod p^{j+1} which is congruent to a mod p^j, (ie., b reduces to a mod p^j, a lifts to b mod p^{j+1}).*

Proof. We're looking for solutions $b = a + tp^j$ where $t \in \{0, 1, \ldots p-1\}$ to the congruence mod p^{j+1}. Want to see if one of these t works. Use Taylor expansion around a:

$$f(a + tp^j) = f(a) + tp^j f'(a) + \frac{(tp^j)^2}{2!} f''(a) + \cdots + \frac{(tp^j)^n}{n!} f^{(n)}(a)$$

Lemma 27.

$$f(a + tp^j) \equiv f(a) + tp^j f'(a) \mod p^{j+1} \text{ if } j \geq 1$$

Proof. $\frac{f^{(n)}}{n!}$ is an integer if f is a polynomial with integer coefficients, and so follows after we see that $p^{nj} \equiv 0 \mod p^{j+1}$. □

(Cont'd next section)

18.781 Theory of Numbers
Spring 2012